I.7 - ORGANIZACIÓN DE UNA COMPUTADORA DIGITAL

Las computadoras no diferencian entre datos e instrucciones, ya que ambos son secuencias de números binarios. El programador es quien define su uso. Las instrucciones son órdenes dadas a la unidad de control para que emita comandos a todas las unidades y se realice una operación específica. Los datos son las cantidades sobre las que se operan.

El funcionamiento de una computadora se basa en la transferencia de información entre diferentes unidades, utilizando registros como soportes de la información. Por lo tanto, se basa en la transferencia de cantidades binarias entre registros, a veces intercalando operadores booleanos.

Esta organización detallada es conocida como la arquitectura de Von Neumann, propuesta inicialmente por John Von Neumann. Todas las computadoras existentes hasta la fecha respetan esta arquitectura, aunque se continúan desarrollando nuevos modelos para aplicaciones específicas.

I.8 - CONFORMACIÓN DE CADA UNA DE LAS UNIDADES COMPONENTES DE UNA COMPUTADORA DIGITAL:

Cada componente de una computadora digital tiene una función específica:

* **Unidad de Entrada**: Conecta la computadora con periféricos para recibir información, reorganizándola y ajustando su velocidad a la de la máquina.
* **Unidad de Salida**: Realiza la función opuesta a la unidad de entrada, adaptando la información procesada para los periféricos destinatarios.
* **Unidad de Memoria**: Almacena datos e instrucciones de manera ordenada y los suministra a la unidad que los solicite.
* **Unidad Lógica y Aritmética (ALU)**: Ejecuta operaciones sobre los datos según las instrucciones del programa, y puede alterar las instrucciones si se le suministran como datos.
* **Unidad de Control**: Interpreta y ejecuta las instrucciones, emitiendo comandos a los componentes de la computadora para realizar la tarea especificada.

Estos componentes trabajan juntos para permitir que la computadora funcione de manera coordinada y eficiente.

I.9 - FORMAS DE FUNCIONAMIENTO DE UNA COMPUTADORA DIGITAL:

En la computación digital, existen dos enfoques principales para la programación:

1. **Programación en Hardware**: Se interconectan circuitos específicos para resolver un problema determinado. Este método es más rápido pero más costoso debido al hardware especializado requerido.
2. **Programación en Software**: En lugar de circuitos fijos, se utilizan circuitos generales por los cuales fluye la información para ser procesada. Este enfoque es más flexible y menos costoso, ya que no depende de hardware especializado.

Ambos métodos buscan resolver problemas, pero la programación en software ofrece una mayor adaptabilidad a diferentes tareas sin la necesidad de cambiar el hardware físico.

I.10 - INTERCONEXIÓN DE LAS UNIDADES DE UNA MÁQUINA.

Podemos visualizar una computadora como un conjunto de unidades interconectadas. En lugar de usar un enredo de cables (método “de todos contra todos”), se prefiere usar un sistema más organizado conocido como **barra ómnibus** o **bus**. Este sistema actúa como una red central por la cual pasan todos los datos y señales, similar a cómo la red eléctrica distribuye energía a las casas. Cada unidad de la computadora se conecta a este bus, lo que permite una comunicación eficiente y ordenada entre las diferentes partes de la computadora.

I.11 - FUNCIONAMIENTO DE UNA COMPUTADORA

El funcionamiento de una computadora se puede resumir en un ciclo de dos fases principales: **búsqueda** y **ejecución**. Durante la fase de búsqueda, la **Unidad de Control** activa los circuitos para localizar una instrucción en la memoria y la transfiere al registro de instrucciones. Luego, en la fase de ejecución, la instrucción se procesa y se llevan a cabo las operaciones necesarias.

Este ciclo se desglosa en ocho subfases:

1. **Cálculo de la Dirección de la Instrucción (CDI)**: Determinar la dirección efectiva de la instrucción.
2. **Búsqueda de la Instrucción (BI)**: Localizar y transferir el código de la instrucción a la Unidad de Control.
3. **Decodificación de la Instrucción (DI)**: Interpretar el código para generar los comandos de ejecución.
4. **Cálculo de la Dirección del Operando (CDO)**: Calcular la dirección del operando referenciado por la instrucción.
5. **Búsqueda del Operando (BO)**: Ubicar y transferir el operando al registro correspondiente.
6. **Ejecución (EJEC)**: Realizar la operación indicada por la instrucción.
7. **Cálculo de la Dirección del Resultado (CDR)**: Calcular la dirección donde se almacenará el resultado.
8. **Almacenamiento del Resultado (AR)**: Guardar el resultado en el lugar adecuado.

Además, se introducen conceptos como **Nueva Instrucción** (la siguiente en el programa), **Lista o Vector** (conjunto ordenado de números), **Operandos Múltiples** (listas o vectores) y **Resultados Múltiples** (varios resultados derivados de una operación).

I.11.1 - INTERRUPCIONES:

Las computadoras tienen un sistema que permite interrumpir las operaciones de la **Unidad Central de Procesamiento (CPU)**, compuesta por la **Unidad de Control** y la **Unidad Lógica Aritmética**. Estas interrupciones mejoran el rendimiento del equipo, ya que los periféricos suelen ser más lentos que la CPU, y sin las interrupciones, la transferencia de datos sería ineficiente, desperdiciando la capacidad total del sistema.

El proceso de interrupciones en una computadora digital se maneja de la siguiente manera:

1. **Interrupciones y Ciclo de Máquina**: Durante una operación de entrada/salida (E/S), que es lenta, la CPU puede seguir ejecutando el programa gracias a un sistema de interrupciones. Si se solicita una operación de E/S y el periférico está disponible, se inicia la transacción. Cuando el buffer de E/S o la memoria del periférico están llenos, se emite una señal de interrupción, devolviendo el control al usuario o al programa. Este ciclo se repite hasta completar la transacción.
2. **Interrupciones Múltiples**: Si durante una interrupción ocurre otra, se pueden manejar de dos maneras:
   * **Deshabilitar nuevas interrupciones**: Mientras se atiende la primera interrupción, se deshabilitan las demás hasta que se resuelva, y luego se atienden las siguientes en orden inverso a su aparición.
   * **Definir prioridades**: Las interrupciones con mayor prioridad se atienden primero, incluso si esto significa interrumpir otras interrupciones de menor prioridad.

Estos métodos permiten que la CPU gestione eficientemente múltiples tareas y solicitudes sin comprometer el rendimiento general del El funcionamiento simplificado de una computadora se puede describir en los siguientes pasos:

1. **Búsqueda de Instrucciones**: La unidad de control busca y trae la próxima instrucción desde la memoria.
2. **Transferencia de Datos**: Se realiza bit a bit, utilizando pulsos de desplazamiento para mover la información de un registro a otro.
3. **Decodificación y Ejecución**: La unidad de control decodifica la instrucción y ejecuta la operación correspondiente, buscando operandos si es necesario y realizando cálculos.
4. **Almacenamiento de Resultados**: Los resultados se almacenan de nuevo en la memoria o se pasan a otros registros.

Este proceso se acompaña de un sistema de interrupciones que permite la entrada y salida de datos, conectando la computadora con dispositivos periféricos y permitiendo la interacción con el programa en ejecución. Las interrupciones pueden ser secuenciales o anidadas, y se manejan para no interrumpir el flujo principal del programa más de lo necesario sistema.

I.12 - FUNCIONAMIENTO SIMPLIFICADO DE UNA COMPUTADORA:

El funcionamiento simplificado de una computadora se puede describir en los siguientes pasos:

1. **Búsqueda de Instrucciones**: La unidad de control busca y trae la próxima instrucción desde la memoria.
2. **Transferencia de Datos**: Se realiza bit a bit, utilizando pulsos de desplazamiento para mover la información de un registro a otro.
3. **Decodificación y Ejecución**: La unidad de control decodifica la instrucción y ejecuta la operación correspondiente, buscando operandos si es necesario y realizando cálculos.
4. **Almacenamiento de Resultados**: Los resultados se almacenan de nuevo en la memoria o se pasan a otros registros.

Este proceso se acompaña de un sistema de interrupciones que permite la entrada y salida de datos, conectando la computadora con dispositivos periféricos y permitiendo la interacción con el programa en ejecución. Las interrupciones pueden ser secuenciales o anidadas, y se manejan para no interrumpir el flujo principal del programa más de lo necesario.

I.12.1 - LENGUAJE DE MÁQUINA:

El **lenguaje de máquina** es la secuencia de ceros y unos con la que operan las computadoras. Escribir y leer en binario es complicado, por lo que se crearon lenguajes más sencillos como el **ensamblador**, que usa caracteres alfanuméricos y es traducido automáticamente a binario por un programa ensamblador.

Luego surgieron **lenguajes de alto nivel** como Cobol, Fortran, Basic, Pascal y C, que son más cercanos al lenguaje humano. Estos pueden ser **compilables** o **interpretables**. Los compilables requieren un compilador que traduce el programa a lenguaje de máquina, mientras que los interpretables se traducen directamente a instrucciones de máquina.

En la programación, al código original se le llama **programa fuente**, y al código resultante en lenguaje de máquina, **programa objeto**.

I.12.3 - SISTEMA OPERATIVO:

El **Sistema Operativo** es un conjunto de programas esenciales para el funcionamiento de una computadora. Permite la interacción entre el usuario y la máquina, gestionando la entrada y salida de datos, la ejecución de aplicaciones y la supervisión del sistema. Los sistemas operativos modernos suelen tener interfaces gráficas que facilitan el uso de la computadora de manera intuitiva.

El sistema operativo se almacena en diferentes tipos de memoria:

* **ROM**: Contiene la información necesaria para iniciar la computadora.
* **RAM**: Almacena los parámetros del sistema y es de bajo consumo.
* **Disco Duro**: Se encarga principalmente de la presentación y gestión de archivos.

Estos componentes trabajan juntos para que la computadora funcione correctamente y sea accesible para los usuarios.

I.13 - SISTEMA DE COMPUTACIÓN:

Un **sistema de computación** no solo incluye la computadora central, sino también una variedad de periféricos que permiten la interacción con el usuario y el procesamiento de datos. Estos periféricos pueden ser pantallas, teclados, lectores de discos, plotters, impresoras, dispositivos de comunicación, sensores, entre otros.

Los sistemas de computación modernos utilizan buses estandarizados como el RS 232 C para comunicaciones y el IEEE 488 para mediciones, facilitando la conexión entre diferentes dispositivos. Los dispositivos de salida más comunes son los monitores CRT y los discos duros magnéticos para almacenamiento. Para el intercambio de programas se usan discos flexibles, para multimedia discos ópticos, y para grandes volúmenes de respaldo, cintas magnéticas.

En entornos de control de procesos o maquinaria, se emplean redes de computadoras para el diseño asistido por computador (CAD) y la manufactura asistida por computador (CAM). Las supercomputadoras actuales, como la serie RS6000 de IBM, están compuestas por cientos de procesadores que trabajan en paralelo, como la famosa “DEEP BLUE” que ganó al campeón mundial de ajedrez en 1997.

1.B.1. REPRESENTACIÓN DE DATOS

1.B.1.1- CODIFICACIÓN: El **código BCD (Decimal Codificado en Binario)** es un sistema de representación numérica que utiliza la base binaria para representar cada dígito de un número decimal de forma individual. En el código BCD, cada dígito decimal es representado por su equivalente en un grupo de cuatro bits binarios. Aquí tienes la representación BCD de los números del 0 al 9:

0 = 0000

1 = 0001

2 = 0010

3 = 0011

4 = 0100

5 = 0101

6 = 0110

7 = 0111

8 = 1000

9 = 1001

Este sistema facilita la interpretación de números decimales en dispositivos digitales y es especialmente útil en aplicaciones donde se requiere una conversión directa y sencilla entre números binarios y decimales.

1.B.1.1.2 - CÓDIGOS POR PESO:

Exactamente, los **códigos por peso** asignan un valor específico a cada posición binaria. En el caso del código BCD, los pesos de izquierda a derecha son **8, 4, 2 y 1**. Para convertir un número binario a su equivalente decimal, se multiplica cada bit por su peso correspondiente y se suman los resultados. Por ejemplo, para el código binario **0110**, la conversión sería:

0 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1 = 0 + 4 + 2 + 0 = 6

Así, el código binario **0110** se traduce al número decimal **6**.

1.B.1.1.3 - CÓDIGOS ALFANUMÉRICOS:

Los **códigos alfanuméricos** son sistemas de codificación que permiten representar números, letras y otros símbolos en una computadora. Dos de los sistemas más conocidos son:

* **EBCDIC**: Utilizado principalmente por IBM, es un código de intercambio de información decimal codificado en binario extendido.
* **ASCII**: Es el Código Estándar Americano para el Intercambio de Información, que utiliza 7 bits para representar 128 caracteres diferentes, incluyendo letras, números y símbolos de control.

En ASCII, los caracteres se representan con un byte (8 bits), aunque solo se usan 7 bits, dejando el octavo bit para otras funciones o para establecer un conjunto de caracteres extendido. Por ejemplo, el número **4** se representa en binario como **0110100**, y el símbolo **+** como **0101011**.

El código ASCII es limitado en cuanto a la cantidad de caracteres que puede representar, por lo que IBM desarrolló el EBCDIC, que tiene espacios adicionales para caracteres o controles nuevos, ampliando las posibilidades de representación en las supercomputadoras.

1.B.1.1.4 - OPERACIONES ENTRE NÚMEROS CODIFICADOS.

Cuando trabajamos con números codificados en BCD y realizamos operaciones como la suma, es importante realizar correcciones para asegurarnos de que el resultado se mantenga dentro del rango válido del código BCD. Aquí tienes un resumen de cómo se manejan las sumas en BCD:

* **Resultado entre 0 y 9**: No se requiere corrección, el resultado es directamente válido en BCD.
* **Resultado entre 10 y 15**: Se debe detectar y corregir, ya que no es un valor válido en BCD. Se agrega 6 al resultado para obtener un número válido en BCD.
* **Resultado de 16 o más**: También requiere corrección. Se agrega 6 a la parte baja del resultado y se lleva 1 a la siguiente posición más significativa.

Por ejemplo, para la suma de **6 + 5** en BCD:

0110 (6)

+ 0101 (5)

------

1011 (Incorrecto en BCD)

+ 0110 (Corrección, agregando 6)

------

1 0001 (Resultado correcto en BCD, que es 11)

Y para **8 + 9**:

1000 (8)

+ 1001 (9)

------

1 0001 (Incorrecto en BCD)

+ 0110 (Corrección, agregando 6)

------

1 0111 (Resultado correcto en BCD, que es 17)

En ambos casos, después de la corrección, obtenemos un resultado válido en BCD que representa correctamente la suma decimal.

1.B.1.2 - NÚMEROS EN COMA FIJA Y EN COMA FLOTANTE:

La **notación de coma fija** mantiene la coma decimal en una posición constante, adecuada para contabilidad. La **notación de coma flotante** es mejor para cifras muy grandes o pequeñas, dividiendo el número en **mantisa** y **exponente**. En computación, los números flotantes se guardan en registros con un bit para el signo y otro para el exponente, permitiendo representar números de forma eficiente.

1.B.1.2.3 - NORMAS DEL IEEE PARA NÚMEROS EN COMA FLOTANTE:

Las **normas IEEE 754** establecen estándares para los números en coma flotante, asegurando consistencia y portabilidad. Los formatos son:

* **Simple**: **32 bits** (1 bit de signo, 8 de exponente, 23 de fracción).
* **Doble**: **64 bits** (1 bit de signo, 11 de exponente, 52 de fracción).

La **variante extendida** ofrece mayor precisión con más bits para exponente y mantisa. El exponente polarizado facilita la representación de un rango amplio de números.

1.B.2 – IMPLEMENTACIÓN DE LAS OPERACIONES LÓGICAS Y ARITMÉTICAS

1.B.2.1 - INTRODUCCION:

El **complemento de un número** es la diferencia entre ese número y un valor base llamado **módulo**. En sistemas binarios, se usa el **complemento a uno** y el **complemento a dos**, donde el complemento a dos es simplemente el complemento a uno más uno. En sistemas decimales, se utilizan el **complemento a nueve** y el **complemento a diez**, siendo este último el complemento a nueve más uno.

1.B.2.2.1 - COMPLEMENTADORES BINARIOS:

Los **complementadores binarios** son dispositivos que invierten los bits de un número binario. El tipo más sencillo es el **serie**, que usa un inversor y un registro de desplazamiento. Existe también el **complementador paralelo**, que realiza la operación de forma simultánea en todos los bits. En la práctica, se emplean **multivibradores** en los registros, que ya proporcionan salidas normales y complementarias, facilitando la obtención del complemento a uno de un número binario.

El **complementador a nueve** se usa para simplificar la resta y división en decimal, restando cada dígito de nueve. Para comparar números, se usan **comparadores** que verifican si dos dígitos binarios son iguales, facilitando la comparación de números decimales codificados en binario al usar varios comparadores en paralelo.

1.B.2.4 - SUMADOR BINARIO:

El **semisumador binario** es un circuito que realiza la suma de dos dígitos binarios. Produce una salida cuando los dígitos son diferentes y no hay salida cuando son iguales. Para el caso de sumar dos unos (1 + 1), el resultado es 10, lo que implica un bit de arrastre. El **sumador completo** es una extensión del semisumador que también maneja este bit de arrastre para sumas de múltiples bits.

1. **Sumador Paralelo con Anticipo del Arrastre**: Para acelerar la suma, se anticipa el arrastre en bloques cortos de sumadores, por ejemplo, de cuatro unidades. De esta manera, el retardo siempre será solamente el correspondiente a la determinación de tres arrastres, pues el cuarto es dado mediante un circuito combinacional.

**Multiplicación de Números Binarios**: La multiplicación entre números binarios puede hacerse fundamentalmente de tres maneras: por sumas sucesivas, acelerada por suma y desplazamiento, y por un sistema matricial o celular.

* + **Multiplicación Binaria por Sumas Sucesivas**: Consiste en sumar tantas veces el multiplicando como lo diga el multiplicador. Se provoca la suma del acumulador y el registro B, previo almacenamiento en él del multiplicando y del multiplicador en un registro auxiliar. Luego se cuenta el número de sumas que se va almacenando en un contador, y se lo compara con el registro auxiliar. Cuando ambos contenidos sean iguales, se detendrá el proceso.
  + **Multiplicación Acelerada**: Cuando haya un 1 en el multiplicador, el multiplicando se sumará y se desplazará, mientras que cuando haya un cero, se desplazará solamente. El registro A, para el resultado, tiene doble longitud, lo cual es lógico pues el producto de dos cantidades tiene una longitud igual al doble de la de cada uno de los operandos. El proceso sigue igual, hasta que se produce el desplazamiento número 4, puesto que esa es la cantidad de bits de los registros, y en ese momento se tendrá el resultado leyendo conjuntamente el contenido de A y A’. Los datos iniciales en decimal eran: 11 x 9 = 99, el resultado es 01100011, lo que significa 1x26 + 1x25 +1x21 +1x20 = 64 + 32 + 2 + 1 = 99, por lo que se ha actuado correctamente.

1. **Multiplicador Celular**: Este es un tipo de multiplicador paralelo en el que se obtiene el producto final directamente en las salidas, con las demoras introducidas por los circuitos solamente. El sistema se basa en sumas y desplazamientos, que son realizados todos por hardware. Es más costoso que los métodos anteriores y solo se justifica en caso de necesitar velocidades de cálculo muy altas.
2. **Mecanismos para la División entre Números Binarios**: La división entre números binarios se realiza de la misma manera que la multiplicación. Puede ser por restas sucesivas, por resta y desplazamiento y mediante un sistema celular. En algunos casos, se puede reemplazar la resta por la suma del complemento.

**División Binaria por Restas Sucesivas**: Este método consiste en restar el divisor del dividendo tantas veces como sea posible, contando el número de restas, lo que será el cociente o resultado de la división. La operación se detiene cuando desaparece el arrastre, lo cual se lleva a cabo mediante una compuerta.

**División Acelerada entre Números Binarios**: Este método consiste en restar y desplazar. Primero, se necesita un registro para el dividendo de doble longitud, que se desplaza totalmente a la izquierda para comenzar. Luego, se prueba si es posible efectuar la resta, es decir, si el contenido del registro acumulador es mayor que el contenido del registro B, estando alineados por su dígito más significativo. Si es posible, se realiza la resta y el primer dígito del cociente será un 1. Si no puede restarse, se desplaza a la izquierda el contenido del acumulador y del cociente, y se vuelve a empezar, en este caso se pondrá un 0 en el lugar correspondiente del cociente. El proceso continúa hasta que la cantidad de desplazamientos sea igual a un número prefijado, donde se dará por terminada la operación. En el acumulador se leerá el resto, y en el registro M, se leerá el cociente. En general, la cantidad de desplazamientos N, será igual a la longitud deseada del cociente. Se proporciona un diagrama de flujo y un esquema del circuito para este método de división binaria acelerada.

**Divisor Celular**: Al igual que en la multiplicación, también es posible realizar la división por restas y desplazamiento mediante un sistema celular, es decir, casi totalmente en paralelo.

**Operaciones entre Números Codificados en Binario**: Se discute la representación de cantidades decimales en binario, específicamente en el código BCD. La representación normal en computadora es mediante el código ASCII, que consiste en un grupo de ocho bits por cada carácter. Sin embargo, para las cifras solo son necesarios cuatro bits, que son los últimos.

**Suma entre Decimales Codificados**: Para representar cada cifra se utiliza un grupo de cuatro binarios. Dado que el código admite más de diez alternativas, exactamente 16, se presentarán errores durante las operaciones, por ejemplo, en la suma de ocho y cinco, cuyo resultado es 13. Este resultado es perfectamente representable para el código, pero no para nosotros, por lo que se necesita corregir estos resultados mayores a nueve y menores que 16. Esta corrección se consigue sumando un seis binario y tomando un arrastre decimal para la próxima suma.